

Research Paper

Pose Fusion AI Framework for Personalized Yoga Asana Guidance Alignment Analysis and Performance Assessment

Dr Swapna Mudrakola, M.Tech, Ph.D

Assistant Professor

Department of Computer Science and Engineering,
Aurora Deemed University, Hyderabad, IndiaSwapna0801@gmail.com

M.L.Sahithi

Student

Department of Computer Science and Engineering,
Aurora Deemed University, Hyderabad, IndiaSahiti.12@gmail.com

Abstract—This paper presents a real-time AI yoga pose instructor that combines person detection, landmark-based geometric representation, temporal deep learning, reference-pose comparison, and web-based feedback. The implemented pipeline detects the practitioner with YOLO11, extracts 33 body landmarks through MediaPipe, converts each valid frame into a 56-dimensional vector containing normalized coordinates, body distances, and joint angles, and forms 30-frame sequences for pose classification. Four temporal and convolutional models—LSTM, GRU, one-dimensional CNN, and ConvLSTM—were trained and evaluated on 43 videos of Tadasana, Trikonasana, and Vrikshasana with video-level train/validation/test separation. The GRU achieved the strongest test performance with 99.31% accuracy, 99.32% precision, 99.31% recall, and 99.31% weighted F1-score on 433 test sequences. Beyond pose identity, the deployed Flask/SocketIO application compares the live pose with reference landmarks, computes landmark and angle similarity, identifies incorrect joints, and provides corrective feedback. A real-time deployment evaluation demonstrates that high pose-recognition confidence can coexist with moderate posture alignment; for example, Tadasana was recognized with 98.66% GRU confidence while posture accuracy was 82.61%, producing a “Needs Improvement” status. The system therefore separates pose recognition from posture-quality assessment and provides interpretable real-time guidance.

Keywords—Yoga pose recognition, GRU, MediaPipe, YOLO11, posture assessment, human pose estimation, real-time feedback.

I. INTRODUCTION

Computer-vision systems for yoga instruction aim to recognize an asana, assess whether the body is aligned with a reference posture, and return feedback quickly enough to be useful during practice. Recent surveys show that yoga-pose recognition has progressed from handcrafted joint-angle rules toward keypoint-based machine learning and deep spatiotemporal models [1], [17]. Systems such as YogNet, deep pose-estimation frameworks, and smart yoga instructors demonstrate that temporal information and skeletal structure can support both recognition and corrective guidance [2]–[6]. However, high pose-classification accuracy alone does not indicate that a pose is geometrically correct. A practitioner may be confidently classified as performing a particular asana while still exhibiting substantial joint misalignment. The framework evaluated in this paper explicitly addresses this distinction. A YOLO11 detector first isolates the person from each video frame. MediaPipe then estimates 33 body landmarks. Instead of feeding raw images directly into the temporal classifier, the implementation constructs 56 compact geometric features: 30 normalized two-dimensional coordinates from 15 selected landmarks, 14 body-distance

ratios, and 12 normalized joint angles. Consecutive observations are arranged into 30-frame sequences, thereby preserving motion continuity and reducing sensitivity to a single unstable frame.

Four architectures were trained under a common protocol: LSTM, GRU, a one-dimensional CNN, and ConvLSTM. The experimental evaluation identifies GRU as the strongest model. The deployed application then adds a second stage that compares current landmarks and angles with a stored reference pose, marks incorrect landmarks, calculates a posture-accuracy score, and generates feedback. The main contributions are therefore: (1) a compact geometric temporal representation for three yoga poses; (2) leakage-resistant video-level evaluation of four deep models; (3) a separate posture-quality score that complements model confidence; and (4) a real-time Flask/SocketIO interface that displays pose, confidence, posture accuracy, status, and joint-level feedback.

II. RELATED WORK

Yoga-pose recognition has been studied using image classification, skeleton features, temporal neural networks, graph models, and interactive feedback systems. Rajendran and Sethuraman [1] provide a broad survey of computer-vision approaches for yogic posture recognition. Yadav et al. [2] proposed YogNet, a two-stream spatiotemporal architecture for real-time multi-person yoga recognition and posture correction. Thoutam et al. [3] combined pose estimation with deep learning and generated feedback for incorrect postures, while Upadhyay et al. [4] developed the Y_PN-MSSD architecture for yoga practitioners. Classical feature-based learning has also remained effective: Palanimeera and Ponmozhi [5] evaluated machine-learning classifiers using joint-angle information.

Real-time guidance is a recurring objective in the literature. Kishore et al. [6] presented a smart instructor that compares captured poses with benchmark postures. Rishan et al. [14] combined OpenPose with temporal CNN/LSTM processing in the Infinity Yoga Tutor, while Gupta and Jangid [18] used pose extraction and machine-learning validation. More recent work has explored graph self-attention [8], ensemble attention models [9], home-based monitoring [10], lightweight visibility-aware networks [11], and accessibility-oriented posture correction [13]. These studies motivate a pipeline in which spatial landmarks, temporal modeling, and corrective feedback are handled as related but distinct tasks.

The proposed framework differs from direct image-classification systems because it uses a detector-to-landmark-

to-feature pipeline before temporal classification. It also avoids sequence leakage by splitting at the source-video level before generating overlapping windows. Most importantly, the front end reports both GRU confidence and a reference-based posture score, preventing the classification probability from being misinterpreted as a measure of biomechanical correctness.

III. MATERIALS AND METHODS

A) Dataset and Video-Level Partitioning

The experimental dataset contains 43 videos distributed across three classes: 15 Tadasana videos, 13 Trikonasana videos, and 15 Vrikshasana videos. During preprocessing, 6,696 Tadasana frames, 5,848 Trikonasana frames, and 7,411 Vrikshasana frames were sampled. After pose detection, 6,359, 5,831, and 7,327 frames, respectively, remained valid. Source videos were split before temporal-window generation: 34 videos for training, four for validation, and five for testing. The video identities in these partitions were verified to be disjoint, which is important because overlapping sequences from the same source would otherwise inflate test performance.

Table I. Dataset and preprocessing summary.

Class	Videos	Sampled	Valid	Failed
Tadasana	15	6696	6359	337
Trikonasana	13	5848	5831	17
Vrikshasana	15	7411	7327	84

B) Person Detection and Landmark Extraction

Each sampled frame is processed by a YOLO11n person detector with a confidence threshold of 0.40. The largest detected person is selected, the bounding box is expanded by 30%, and the crop is letterbox-resized to 512×512 pixels. MediaPipe Pose then estimates 33 landmarks represented by image coordinates and confidence-related fields. Frames for which the person or pose is not detected are excluded from feature generation. This detector-first design reduces background variation and gives MediaPipe a normalized person-centered region [22]–[24].

C) Geometric Feature Engineering and Temporal Sequences

The 33-landmark pose is transformed into a 56-dimensional frame descriptor. Fifteen landmarks—nose, both ears, shoulders, elbows, wrists, hips, knees, and ankles—contribute 30 normalized x-y coordinates. Translation is controlled using the hip center, while a shoulder-based body scale reduces sensitivity to camera distance. Fourteen normalized Euclidean distances encode limb and torso geometry. Twelve joint angles are calculated from landmark triplets and divided by 180 so that they lie approximately in the interval [0,1]. For temporal modeling, 30 consecutive feature vectors form one sequence and adjacent sequences are generated with stride five. The final sequence arrays contain 2,849 training samples, 395 validation samples, and 433 test samples, each with shape 30×56 . StandardScaler is fitted only on the flattened training data and then applied to validation and test sequences, preventing test information from influencing normalization.

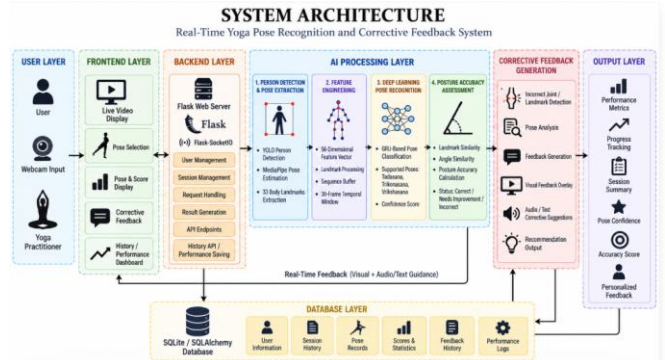


Fig. 1. Proposed system architecture for real-time yoga pose recognition and posture assessment.

D) Deep Learning Models

The LSTM and GRU models use two recurrent layers with hidden size 128, dropout 0.30, and a final three-class linear layer. The one-dimensional CNN applies two temporal convolution stages ($56 \rightarrow 64$ and $64 \rightarrow 128$ channels), batch normalization, ReLU activation, max pooling, adaptive average pooling, dropout, and a three-class output. The ConvLSTM reshapes the 56-dimensional vector into a 7×8 feature map and processes the sequence with a convolutional recurrent cell before global spatial averaging. All models use cross-entropy loss and Adam optimization with learning rate 0.001. The training configuration uses batch size 64 and a maximum of 50 epochs, and the checkpoint with the strongest validation accuracy is retained.

E) Performance Metrics

Accuracy: Accuracy measures the proportion of all evaluated samples that are classified correctly. For a one-versus-rest interpretation of each yoga class, it is calculated as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: Precision indicates how many samples predicted as a particular pose actually belong to that pose. It penalizes false positive pose assignments and is expressed as

$$Precision = \frac{TP}{TP + FP}$$

Recall: Recall measures the proportion of true samples of a pose that are successfully detected. It is important when missed instances of an asana should be minimized and is defined as

$$Recall = \frac{TP}{TP + FN}$$

F1-Score: The F1-score is the harmonic mean of Precision and Recall and summarizes their balance using

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Weighted evaluation: Because the test-set supports differ across Tadasana, Trikonasana, and Vrikshasana, weighted Precision, Recall, and F1-score are reported. Each class metric is weighted by its support so that the overall score reflects the empirical class distribution while preserving class-specific evaluation.

F) Reference-Based Posture Accuracy and Feedback

Recognition confidence and posture accuracy are computed independently in the deployed application. Once the sequence buffer is ready, GRU predicts the pose and confidence. For the selected target asana, the current MediaPipe landmarks are compared with reference landmarks. Landmark similarity and angle similarity are combined by the posture-accuracy module; joint evaluation identifies incorrect landmarks, and a

feedback module converts these differences into corrective messages. The final posture score is clipped to 0–100%. Scores of at least 85% are labeled “Correct,” scores from 70% to below 85% are labeled “Needs Improvement,” and lower scores are labeled “Incorrect.” Correct landmarks are drawn in green and incorrect landmarks in red.

G) Web Deployment

The application is implemented with Flask and Flask-SocketIO for low-latency frame exchange. The client sends webcam frames as encoded images, while the server performs detection, landmark extraction, feature generation, sequence buffering, GRU inference, reference comparison, feedback generation, and annotated-frame rendering. The response returns detected pose, selected pose, GRU confidence, landmark score, angle score, posture accuracy, status, incorrect landmarks, feedback, and the rendered frame. SQLite/Flask-SQLAlchemy stores performance history containing pose, posture accuracy, confidence, and time information.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

A) Comparative Model Performance

Table II summarizes the test results obtained during model evaluation. GRU clearly provides the strongest balance across all four classification metrics, with 99.31% accuracy, 99.32% precision, 99.31% recall, and 99.31% F1-score. CNN achieves 93.30% accuracy and LSTM reaches 92.38%, whereas ConvLSTM obtains 89.61%. GRU also records the lowest test loss (0.0227) and a best validation accuracy of 99.49%. These results indicate that gated temporal modeling is well matched to the compact 56-feature sequence representation.

Table II. Test performance of the evaluated deep learning models.

Model	Loss	Accuracy	Precision	Recall	F1
LSTM	0.4711	92.38	94.48	92.38	92.65
GRU	0.0227	99.31	99.32	99.31	99.31
CNN	0.1647	93.30	94.97	93.30	93.52
ConvLSTM	0.3276	89.61	90.62	89.61	89.56

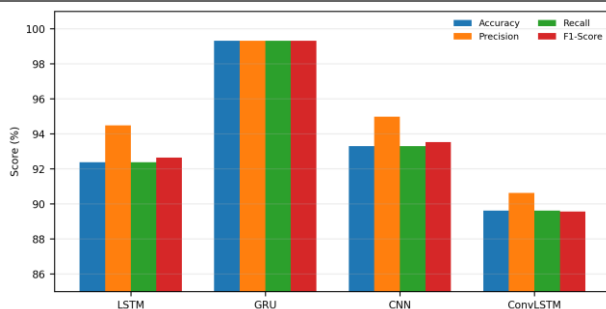


Fig. 2. Combined Accuracy, Precision, Recall, and F1-score comparison.

B) Class-Wise GRU Evaluation

The best GRU model is also consistent across individual poses. Tadasana achieves 100% recall with 97.50% precision and 98.73% F1-score. Trikonasana obtains 100% precision, 99.43% recall, and 99.72% F1-score. Vrikshasana achieves 99.44% precision, 98.88% recall, and 99.16% F1-score. On 433 test sequences, only three predictions are incorrect: one Trikonasana sample is classified as Vrikshasana and two Vrikshasana samples are classified as Tadasana. This class-level result supports the use of GRU as the deployed classifier rather than selecting a model only from aggregate accuracy.

Table III. Class-wise test performance of the selected GRU model.

Pose	Precision	Recall	F1	Support
Tadasana	97.50	100.00	98.73	78

Trikonasana	100.00	99.43	99.72	176
Vrikshasana	99.44	98.88	99.16	179

C) Training Dynamics and Checkpoint Selection

Model selection is based on validation performance rather than the final training epoch. This distinction is important because all four networks eventually fit the training sequences very strongly, while validation behavior differs. The best validation accuracy recorded for GRU is 99.49%, compared with 95.44% for CNN, 93.42% for ConvLSTM, and approximately 91.90% for LSTM. The selected checkpoint is therefore the saved state with the highest validation accuracy, and the untouched test set is evaluated only after checkpoint selection.

Table IV. Best validation accuracy and final test loss used for model selection analysis.

Model	Best Val. Acc. (%)	Test Loss
LSTM	91.90	0.4711
GRU	99.49	0.0227
CNN	95.44	0.1647
ConvLSTM	93.42	0.3276

The GRU learning trajectory shows rapid improvement during the early epochs followed by stable validation accuracy close to 99.5%. Training accuracy reaches 100%, while validation loss remains low around the best checkpoint. This pattern supports the use of validation-based checkpointing rather than simply retaining the last epoch. It also explains why the test loss of GRU is substantially lower than that of the other models.

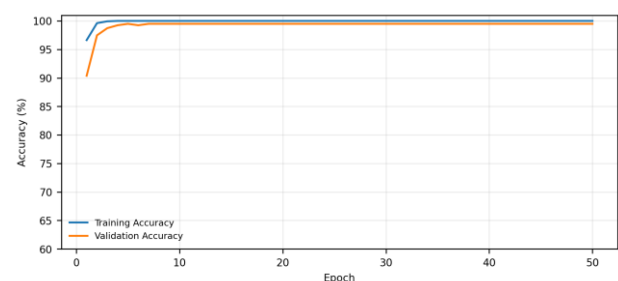


Fig. 3. GRU training and validation accuracy across the 50-epoch training process.

D) Front-End Evaluation Results

The real-time deployment evaluation demonstrates the complete operational workflow. In the first case, the user selects Tadasana and follows the displayed reference pose while the webcam stream is analyzed in real time. The interface reports the detected pose as Tadasana with 98.99% GRU confidence and a posture accuracy of 79.77%, resulting in the status “Needs Improvement.” This illustrates that correct pose identity does not automatically imply accurate alignment.

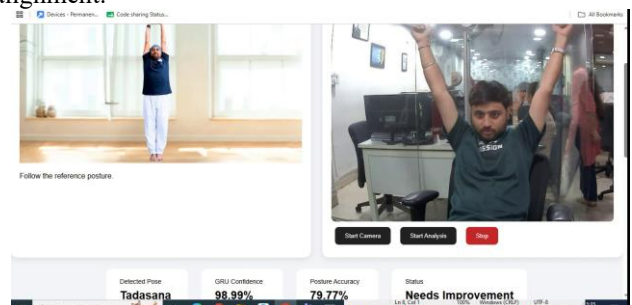


Fig. 4. Tadasana practice interface showing 98.99% GRU confidence and 79.77% posture accuracy.

A later frame exposes the detailed Pose Analysis panel. Tadasana is recognized with 98.66% confidence, while the independent posture score is 82.61%; the system again returns

“Needs Improvement.” Red and green landmark overlays visually localize body regions that deviate from the reference. The small difference between classification confidence and posture score is expected because GRU answers “which pose is being performed?” whereas reference similarity answers “how closely is the pose aligned?”

Tadasana	98.66%
----------	--------

Pose Analysis

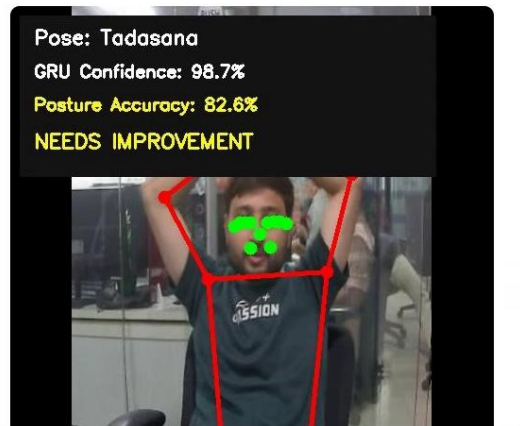


Fig. 5. Pose-analysis interface with landmark overlay and corrective status.

E) Posture Quality Versus Classification Confidence

The deployment results reveal an important design property. Model confidence should not be used as a substitute for posture correctness. A sequence may contain the characteristic temporal and skeletal pattern of Tadasana, giving GRU a confidence near 99%, even though shoulder, elbow, wrist, hip, or leg geometry differs from the stored reference. The reference-comparison stage therefore protects the user interface from presenting a high classification probability as a high-quality pose score. This separation is especially valuable for instruction systems, where recognition and correction are distinct tasks.

The status thresholds further convert the continuous posture score into simple guidance. A score above 85% indicates adequate alignment, 70–85% signals that the pose is recognizable but requires correction, and a score below 70% is marked incorrect. Because the server also returns the list of incorrect landmarks and generated feedback, the interface can direct the user toward specific joints instead of presenting a single unexplained number.

Table V. Representative front-end inference results.

Case	Target/Detected	GRU Conf.	Posture Acc.	Status
1	Tadasana / Tadasana	98.99	79.77	Needs Improvement
2	Tadasana / Tadasana	98.66	82.61	Needs Improvement
3	Tadasana / Tadasana	97.39	31.71	Incorrect

F) Sequence-Level Generalization and Leakage Control

The experimental design deliberately generates windows only after the video-level split. This is a critical methodological choice for temporal pose data. With a sequence length of 30 and stride five, neighboring windows share 25 frames and are therefore highly correlated. If those windows were randomly divided after generation, near-duplicate temporal content from a single source sequence could enter both training and test sets. Verification of the partitioning confirms that the

intersections among training, validation, and test video identifiers are empty before scaling and training.

The resulting test set contains 433 sequences derived from five videos that were never used for optimization. Although this is still a small number of source recordings, the protocol is more conservative than frame-level splitting. The same principle is applied to feature scaling: StandardScaler is fitted on 85,470 training-frame feature vectors obtained by reshaping the training sequences and is then reused without refitting on validation or test data. These controls reduce two common forms of information leakage in pose-recognition experiments.

G) Deployment Behavior and Computational Design

The deployed application uses an independent sequence buffer for each yoga class and waits until 30 valid feature frames are available before GRU inference. While the buffer is filling, the front end receives progress information rather than a premature classification. A non-blocking frame lock drops incoming frames when the previous frame is still being processed, preventing request queues from growing during live webcam use. This is a practical engineering choice for interactive systems because bounded latency is more useful than processing every captured frame with increasing delay.

The server validates the availability of YOLO, MediaPipe, and GRU before processing. If no person is found, it returns a “Person not detected” state; if MediaPipe cannot estimate the skeleton, it returns “Pose not detected.” Once a valid 30-frame sequence is obtained, the response includes pose identity, confidence, posture accuracy, landmark and angle scores, incorrect landmarks, feedback, and an annotated JPEG frame. This explicit error handling makes the application behavior easier to interpret than a single silent prediction endpoint.

Performance records can be saved to the database with pose, posture-accuracy score, and confidence. The administrative view aggregates total detections and average accuracy by pose, while the history page provides session-level records. These components are not part of the classifier itself, but they make the prototype suitable for monitoring repeated practice and for examining whether a user’s posture score improves over time.

H) Error Characteristics and Feedback Interpretation

The GRU class report shows that the remaining classification errors are highly concentrated rather than distributed across every class. Tadasana has perfect recall; one Trikonasana sequence is assigned to Vrikshasana, and two Vrikshasana sequences are assigned to Tadasana. Because Vrikshasana shares an upright torso with Tadasana but differs strongly in leg configuration, these errors suggest that temporally unstable leg landmarks or transition frames can influence the sequence representation. Future models could give greater weight to lower-body joint reliability or use attention over the 56 features.

Front-end posture errors are intentionally treated differently. In a representative low-quality execution, Tadasana still receives 97.39% recognition confidence while posture accuracy falls to 31.71% and the status becomes “Incorrect.” This behavior is desirable: the pose recognizer identifies the semantic activity, while the reference module independently judges execution quality. A correction system that merged these quantities into one score could incorrectly reassure the user whenever the class label was obvious.

I) Discussion

The comparative results suggest that sequence modeling is more important than model complexity for this dataset. GRU substantially outperforms the tested ConvLSTM even though ConvLSTM has a more elaborate spatial recurrent operation. The engineered features already encode body geometry, so forcing the 56-element vector into a synthetic 7×8 spatial grid may provide less useful inductive structure than directly learning temporal dependencies. The one-dimensional CNN and LSTM remain competitive but show more confusion between Vrikshasana and Tadasana than GRU.

Video-level partitioning is another strength of the experiment. Since sequences are generated with stride five from the same recordings, a random split performed after window generation would place highly similar neighboring windows in both training and test sets. By separating source videos before creating sequences, the reported accuracy more closely reflects generalization to unseen recordings. Standardization is likewise fitted only on the training partition.

J) Limitations and Practical Considerations

The current experiment includes only three asanas and 43 source videos; therefore, the 99.31% sequence-level GRU result should not be interpreted as a universal yoga-recognition benchmark. Broader evaluation should include more practitioners, camera viewpoints, clothing conditions, lighting, occlusion, body proportions, and a larger vocabulary of visually similar poses. The reference-based posture score also depends on the quality and representativeness of the stored reference landmarks. A single reference may not capture safe anatomical variation among users.

Future extensions can learn personalized reference ranges, use multiple expert examples per pose, estimate uncertainty, and aggregate posture scores over time rather than responding to isolated frames. More detailed biomechanical feedback could weight clinically important joints differently and adapt angle tolerances to height, mobility, or experience level. Cross-subject and cross-environment evaluation would provide a stronger measure of deployment robustness.

K) Real-Time Interaction and Output Contract

The live-analysis component is organized around a clear response contract between the SocketIO server and browser. Before inference, the client specifies the selected target pose and streams encoded webcam frames. The server first checks that the YOLO detector, MediaPipe pose detector, and GRU inference object are available. If any component is unavailable, an explicit error message is returned instead of silently producing a prediction. When a valid person is detected but the temporal buffer has not yet accumulated 30 feature vectors, the response contains the number of collected frames, the required sequence length, and a percentage progress value. This allows the interface to distinguish “model not ready” from “pose not detected.”

After the sequence is ready, the response becomes substantially richer. It contains the predicted pose, the pose selected by the user, GRU confidence, landmark-similarity score, angle-similarity score, final posture accuracy, qualitative status, the list of incorrect landmarks, generated feedback, and an annotated frame. These values are produced from two related paths: the GRU path recognizes the temporal class, while the reference path assesses geometric quality. The separation makes the front-end result auditable because the user can see both the classification decision and the posture-correction evidence.

Table VI. Main real-time outputs returned by the deployed yoga-analysis service.

Output	Interpretation
Pose	GRU-predicted yoga class
Confidence	Probability/confidence of pose identity
Landmark score	Similarity of current and reference landmarks
Angle score	Similarity of selected joint angles
Posture accuracy	Combined alignment score (0–100%)
Status	Correct / Needs Improvement / Incorrect
Incorrect landmarks	Body joints requiring attention
Feedback	Human-readable corrective messages

L) Session History, User Profile, and Administrative Monitoring

The implementation also includes persistent user and performance management. A user can register, sign in, maintain profile information, select a target pose, start or stop a live session, review practice history, and save performance results. The stored performance record contains the pose name, posture-accuracy value, GRU confidence, and creation time. This makes it possible to inspect changes across repeated sessions instead of treating every prediction as an isolated demonstration.

The administrative dashboard aggregates registered users, total detections, login activity, average posture accuracy, per-pose detection counts, and per-pose average accuracy. These summaries are useful for prototype evaluation because they expose how often the system is used and whether some poses systematically produce lower scores. The database layer is therefore not part of model training, but it supports longitudinal observation and application-level quality assurance. In a larger study, the same mechanism could be extended to anonymized subject identifiers and session-level statistics for cross-user validation.

M) Reliability, Scope, and Responsible Interpretation

Several implementation safeguards improve reliability during interactive use. Incoming frames are decoded defensively; invalid base64 data or undecodable images do not enter the pose pipeline. Person absence and pose-estimation failure are returned as separate states. Session start clears the corresponding temporal buffer so that feature vectors from a previous attempt are not mixed with a new attempt, and the reset endpoint can clear all buffers when required. Database updates are committed transactionally, with rollback on failure. These software checks complement the statistical controls used during model evaluation.

The prototype should nevertheless be interpreted as an exercise-assistance system rather than a medical or rehabilitation device. Its posture score is a geometric similarity measure relative to a stored reference and does not establish whether a movement is clinically safe for every individual. Users with injuries, mobility limitations, or rehabilitation requirements may require professional supervision and personalized range-of-motion limits. In research deployment, feedback thresholds should therefore be validated with qualified instructors and with participants representing different body types and movement capabilities. A stronger future evaluation would report subject-independent results, per-video rather than only per-sequence metrics, confidence calibration, latency on ordinary CPU hardware, and agreement between the automated posture

score and expert human ratings. The current results demonstrate strong performance of the implemented three-pose pipeline on the evaluated dataset and show that the interface can distinguish pose identity from execution quality; broader claims should be supported by larger and more diverse validation data.

V. CONCLUSION

This paper documented and evaluated a complete real-time yoga-pose recognition and posture-assessment system through the proposed implementation and experimental evaluation. The processing pipeline combines YOLO11 person localization, MediaPipe 33-landmark extraction, 56 geometric features, 30-frame temporal sequences, and comparative deep learning. Among LSTM, GRU, CNN, and ConvLSTM, GRU achieved the best test results: 99.31% accuracy, 99.32% precision, 99.31% recall, and 99.31% weighted F1-score. Class-wise scores remain above 98.7% F1 for all three asanas.

The deployed system extends classification into actionable guidance by comparing current landmarks and angles with a reference pose. Front-end evaluation results confirm that high GRU confidence and lower posture accuracy can occur simultaneously, which validates the decision to separate pose identity from posture quality. The resulting architecture is suitable as a compact AI yoga instructor for real-time practice, while future work should enlarge the asana set, increase participant diversity, personalize posture tolerances, and conduct cross-user validation.

REFERENCES

- [1] A. K. Rajendran and S. C. Sethuraman, "A survey on yogic posture recognition," *IEEE Access*, vol. 11, pp. 11183–11223, 2023, doi: 10.1109/ACCESS.2023.3240769.
- [2] S. K. Yadav, A. Agarwal, A. Kumar, K. Tiwari, H. M. Pandey, and S. A. Akbar, "YogNet: A two-stream network for realtime multiperson yoga action recognition and posture correction," *Knowl.-Based Syst.*, vol. 250, Art. no. 109097, 2022, doi: 10.1016/j.knosys.2022.109097.
- [3] V. A. Thoutam, A. Srivastava, T. Badal, V. K. Mishra, G. R. Sinha, A. Sakalle, H. Bhardwaj, and M. Raj, "Yoga pose estimation and feedback generation using deep learning," *Comput. Intell. Neurosci.*, vol. 2022, Art. no. 4311350, 2022, doi: 10.1155/2022/4311350.
- [4] A. Upadhyay, N. K. Basha, and B. Ananthakrishnan, "Deep learning-based yoga posture recognition using the Y_PN-MSSD model for yoga practitioners," *Healthcare*, vol. 11, no. 4, Art. no. 609, 2023, doi: 10.3390/healthcare11040609.
- [5] J. Palanimeera and K. Ponmozhi, "Classification of yoga pose using machine learning techniques," *Mater. Today Proc.*, vol. 37, pt. 2, pp. 2930–2933, 2021, doi: 10.1016/j.matpr.2020.08.700.
- [6] D. M. Kishore, S. Bindu, and N. K. Manjunath, "Smart yoga instructor for guiding and correcting yoga postures in real time," *Int. J. Yoga*, vol. 15, no. 3, pp. 254–261, 2022, doi: 10.4103/ijoy.ijoy_137_22.
- [7] A. S. Talaat, "Novel deep learning models for yoga pose estimator," *SN Appl. Sci.*, vol. 5, Art. no. 341, 2023, doi: 10.1007/s42452-023-05581-8.
- [8] G. Wei, H. Zhou, L. Zhang, and J. Wang, "Spatial-temporal self-attention enhanced graph convolutional networks for fitness yoga action recognition," *Sensors*, vol. 23, no. 10, Art. no. 4741, 2023, doi: 10.3390/s23104741.
- [9] Y. Kou and H. Li, "Image-based fitness yoga pose recognition: Using ensemble learning and multi-head attention," *Int. J. Comput. Intell. Syst.*, vol. 17, Art. no. 257, 2024, doi: 10.1007/s44196-024-00662-x.
- [10] S.-C. Yeh and C.-K. Yang, "Yoga pose recognition and motion analysis for a home-based fitness monitoring and health management system," *Signal Image Video Process.*, vol. 19, Art. no. 841, 2025, doi: 10.1007/s11760-025-04436-6.
- [11] L. Thushara, P. Abdul Jabbar, and K. P. Pushpalatha, "VisYogaNet: Visibility-guided lightweight network for yoga pose recognition," *Int. J. Data Sci. Anal.*, vol. 22, Art. no. 179, 2026, doi: 10.1007/s41060-026-01153-2.
- [12] A. Sharma, V. Sharma, P. Sharma, R. Ladha, Y. Lamba, and S. Bansal, "Automated identification of Shankha Prakshalana yoga poses with machine learning techniques," *Sci. Rep.*, vol. 16, Art. no. 430, 2026, doi: 10.1038/s41598-025-29984-2.
- [13] V. Aggarwal, V. Yadav, S. Chaudhary, G. Singal, R. Kushwaha, and L. K. Sharma, "Real-time yoga posture correction using deep learning for individuals with physical disabilities," *Sci. Rep.*, 2026, doi: 10.1038/s41598-026-54280-y.
- [14] F. Rishan, B. De Silva, S. Alawathugoda, S. Nijabdeen, L. Rupasinghe, and C. Liyanapathirana, "Infinity yoga tutor: Yoga posture detection and correction system," in *Proc. 5th Int. Conf. Inf. Technol. Res. (ICITR)*, 2020, pp. 1–6, doi: 10.1109/ICITR51448.2020.9310832.
- [15] Y. Wanjun, C. Chong, and C. Rui, "Yoga action recognition based on STF-ResNet," in *Proc. IEEE 3rd Int. Conf. Power, Electron. Comput. Appl. (ICPECA)*, 2023, pp. 556–560, doi: 10.1109/ICPECA56706.2023.10076099.
- [16] P. Sarma and S. I. Singh, "Handcrafted feature assisted light-weight encoder decoder based classifier for yoga posture recognition," in *Proc. 2nd Int. Conf. Device Intell., Comput. Commun. Technol. (DICCT)*, 2024, doi: 10.1109/DICCT61038.2024.10532964.
- [17] V. Chamola, E. P. Gummana, A. Madan, B. K. Rout, and J. J. P. C. Rodrigues, "Advancements in yoga pose estimation using artificial intelligence: A survey," *Current Bioinformatics*, vol. 19, no. 3, pp. 264–280, 2024, doi: 10.2174/1574893618666230508105440.
- [18] A. Gupta and A. Jangid, "Yoga pose detection and validation," in *Proc. Int. Symp. Asian Control Assoc. Intell. Robot. Ind. Autom. (IRIA)*, 2021, pp. 319–324, doi: 10.1109/IRIA53009.2021.9588714.
- [19] V. C. R. Pala, S. Kamatagi, S. Jangiti, K. Swaraja, K. R. Madhavi, and G. S. N. Kumar, "Yoga pose recognition with real time correction using deep learning," in *Proc. Int. Conf. Sustainable Comput. Data Commun. Syst. (ICSCDS)*, 2023, pp. 387–393, doi: 10.1109/ICSCDS56580.2023.10104781.
- [20] H. C. Saini, "iSmartYog: A real time yoga pose recognition and correction feedback model using deep learning for smart healthcare," in *Proc. Int. Conf. Smart Syst. Appl. Electr. Sci. (ICSSSES)*, 2023, pp. 1–6, doi: 10.1109/ICSSSES58299.2023.10201061.
- [21] Y.-H. Lo, C.-C. Yang, H. Ho, and S.-W. Sun, "richYoga: An interactive yoga recognition system based on rich skeletal joints," in *Proc. IEEE Int. Conf. Artificial Intell. Virtual Reality (AIVR)*, 2021, pp. 256–257.
- [22] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "BlazePose: On-device real-time body pose tracking," *arXiv:2006.10204*, 2020.
- [23] C. Lugaesi et al., "MediaPipe: A framework for building perception pipelines," *arXiv:1906.08172*, 2019.
- [24] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 172–186, 2021.
- [25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [26] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, 2014, pp. 1724–1734.
- [27] X. Shi et al., "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 802–810.

- [28] A. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” in *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
- [29] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788.